Simulation Study of a GCF Retransmission Scheme

L. R. Welch[1]
Communications Systems Research Section

In this article, we study a promising retransmission algorithm for correcting GCF errors, using both actual GCF 4.8 kbps error data and Adeyemi's model. The results indicate that virtually all GCF error bursts can be corrected with a fairly simple scheme.

I. Introduction

In the Ground Communication Facility (GCF), data are transmitted in 1200-bit blocks. While 99% of the blocks are error-free, 1% have errors, and these tend to bunch together in small groups. Since parity bits within the block can be used to detect block errors with a conditional error rate of less than 10^{-6}, the receiver knows which blocks are in error and can request retransmission. In this article, we analyze one possible retransmission algorithm.

Three types of data must be stored. Since the receiver must pass the blocks on in proper sequential order, it is necessary to store blocks correctly received until all prior blocks have also been correctly received. This will be called the receiver buffer. When the transmitter transmits a block, it must retain that block until correct reception has been verified. This will be called the transmit storage. Finally, data entering the transmitter while other blocks are being retransmitted must be stored. This will be called the transmitter buffer. The transmit storage and transmitter buffer could be part of the same physical device but are treated separately in this simulation.

The operating procedure is as follows. When a data block arrives at the transmitter, it is placed in the buffer. If all error indications have been acted on, the oldest block in the transmitter buffer is transmitted and simultaneously placed in transmit storage. When an error indication is received, the corresponding transmit block is retransmitted and reinserted in transmit storage at the first available block time. The receiver checks the parity bits and sends to the transmitter the information as to whether the block has been received correctly[2]. The receiver stores only those blocks which have been received correctly but for which prior blocks have not yet been received correctly.

[1]Consultant, University of Southern California.

[2]This is done by sending an acknowledgment signal for blocks received without errors, so that a missing acknowledgment means an error has occurred. This technique prevents disaster if the feedback channel is in a noisy condition.

II. Simulation Model

The basic unit of time in the model is the time to transmit one block. For each unit of time, 1200 bits of channel data are examined to determine whether a block error has occurred. The types of data used are discussed later. There are two parameters in the model, NR and LD. The first is an integer describing the input data rate: for every NR transmitted-block time, $NR - 1$ data blocks enter the system at times $t \neq 0 \bmod NR$. The choices in this simulation study were $NR = 10$ and 20. This corresponds to data rates of 0.9 and 0.95 data blocks per channel block. The parameter LD, the loop delay, is the delay from the first transmission to the first opportunity to retransmit after an error has been detected. For the 4.8-kbps simulation, this was taken as 5 block times, and for the 7.2-kbps simulation as 10 block times.

Since the transmitter buffer operates on a first-in, first-out principle, the only information needed to indicate its status is the identification number of the oldest and youngest data blocks in the buffer.

The transmit storage operation was described in the Introduction. There are LD block storage locations and LD indicator bits. The block transmitted at time t is also stored at location $t\,(\mathrm{mod}\ LD)$. Some time prior to $t + LD$, the transmitter receives a verification or rejection message and appropriately sets the $t\,(\mathrm{mod}\ LD)$ indicator bit. Since $t + LD\,(\mathrm{mod}\ LD) = t\,(\mathrm{mod}\ LD)$, a retransmitted block is reinserted into the same storage location. When the transmitter buffer is empty and the indicator bit indicates no error, a "blank" must be sent. In this simulation, we assume that blanks received in error are not retransmitted.

The receiver status is more complicated. One could assume that the receiver stores only good blocks, but this leads to some data shuffling or address indexing problems in certain situations. A simple scheme is as follows. If the receiver has correctly received all data blocks with indices less than or equal to k but has not correctly received $k + 1$, and the largest index of a correct block is $k + \ell$, then each correct block with index n, such that $k < n \leq k + \ell$, is stored in address $n - k$. With this procedure, when a missed block is corrected before $k + 1$ block, it can be stored in appropriate order. Also, if a block is missed but its index can be determined, it can still be stored in correct position as a hedge against disaster. At any instant of time, the storage allocation is the ℓ in the above description.

III. Channel Data

For the 4.8-kbps simulations, 29 real data runs obtained by McClure (Ref. 1) were used to provide data patterns. In addition, the Adeyemi model (Ref. 2) was used to generate 124 runs of channel simulations. Half of these used Adeyemi parameters obtained from the worst real data run and the other half used model parameters derived from a combination of the worst three runs. The random data generator generated a run until 5000 errors occurred. The total bit lengths of the runs varied from 6×10^6 to 2.5×10^7 bits.

Since no 7.2-kbps data were available, only Adeyemi's model was used. The parameters referred to above were modified to increase the expected time in the good state by a factor of $3/2 = (7.2/4.8)$, and to increase the length of garbled transmissions by a similar factor without significantly changing the statistics within the garbled segments. There were 31 runs for each of two parameter sets.

IV. Results

A general conclusion obtained from this study is that the transmitter buffer is much larger than the receiver buffer. This is the case because, when garbled stretches are short, the Receiver Buffer builds up to $LD - 1$ and then quickly recovers, while if garbled stretches are long, the channel becomes busy with retransmissions and data blocks with higher indices build up in the transmitter buffer.

The largest receiver buffer for 4.8-kbps ($LD = 5$) usage was 11 blocks, and the largest for 7.2-kbps ($LD = 10$) usage was 18. This suggests that $2 \times LD$ is the largest size the buffer reaches. A straightforward analysis shows that the transmitter buffer buildup is insensitive to LD but depends strongly on channel statistics.

Neglecting the very bad first real data run, the largest transmitter buffer used was 20 blocks for the 4.8-kbps channel and a data rate of 0.9; the buffer size increases to 25 when the data rate is 0.95. For the 7.2-kbps channel, the largest transmitter buffer required was 23 blocks at the 0.9 rate and 30 blocks at the 0.95 rate.

The results are summarized in Tables 1 and 2. The author has available run-by-run buffer size data for the reported simulations and has the program to generate more simulations if desired.

References

1. McClure, J. P., "4800 bps High Speed Data Error Statistics," JPL IOM, January 5, 1973 (JPL internal document).

2. Adeyemi, O., *Error Control in the GCF: An Information-Theoretic Model for Error Analysis and Coding*, JPL Technical Memorandum (in preparation).

Table 1. Maximum buffer usage for 4.8-kbps channel (measured in 1200-bit blocks)

Data type	No. of runs	Info. rate	Largest (and second-largest) Trans. Buffer	Largest (and second-largest) Rec. Buffer
Real	29	0.9	162 (14)	12 (9)
Adeyemi (parameters from worst run)	62	0.9	20 (19)	11 (10)
Adeyemi (parameters from worst three runs)	62	0.9	17 (16)	9 (8)
Adeyemi (model of worst run)	31	0.95	25 (22)	10 (9)
Adeyemi (model of worst three runs)	31	0.95	19 (16)	9 (8)

Table 2. Maximum buffer usage for 7.2-kbps channel (measured in 1200-bit blocks)

Data type	No. of runs	Info. rate	Largest (and second-largest) Trans. Buffer	Largest (and second-largest) Rec. Buffer
Adeyemi (model of worst run)	31	0.9	21 (19)	18 (18)
Adeyemi (model of worst three runs)	31	0.9	23 (22)	17 (13)
Adeyemi (model of worst run)	31	0.95	30 (25)	17 (17)
Adeyemi (model of worst three runs)	31	0.95	25 (24)	17 (13)